View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by DSpace at Belgorod State University

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

РАЗРАБОТКА ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА КОДИРОВААНИЯ ЦИФРОВЫХ ДАННЫХ С ПОМЩЬЮ ГЕНЕТИЧЕСКОГО АЛГОРИТМА ДЛЯ МИКРОКОНТРОЛЛЕРА ATMEGA 328

Выпускная квалификационная работа
обучающегося по направлению подготовки 09.03.02 Информационные системы и технологии
очной формы обучения, группы 12001508
Набока Георгия Константиновича

Научный руководитель
к.т.н., доцент
Титов А.И.

БЕЛГОРОД 2019

РЕФЕРАТ

Разработка программно-аппаратного комплекса кодирования цифровых данных с помощью генетического алгоритма для микроконтроллера Atmega 328. – Набока Георгий Константинович, выпуская квалификационная работа бакалавра Белгород, Белгородский государственный национальный исследовательский университет (НИУ «БелГУ»), количество страниц 57, включая приложения 64, количество рисунков 20, количество таблиц 5, количество использованных источников 36.

КЛЮЧЕВЫЕ СЛОВА: микроконтроллер, скрещивание, генетический алгоритм, эволюционные методы, кодирование данных, формирование потомка, мутация.

ОБЪЕКТ ИССЛЕДОВАНИЯ: реализация генетического алгоритма на микроконтроллере Atmega 328

ПРЕДМЕТ ИССЛЕДОВАНИЯ: кодирование данных эволюционными методами

ЦЕЛЬ РАБОТЫ: исключение из процесса шифрования данных на внешнем носителе необходимости применения ПК

ЗАДАЧИ ИССЛЕДОВАНИЯ: провести анализ подходящих микроконтроллеров и методик кодирования данных с помощью генетического алгоритма; разработать структурную схему программно-аппаратного комплекса; разработать программное обеспечение низкого уровня для микроконтроллера, входящего в состав комплекса; исследовать эффективность работы реализованного метода кодирования данных в разработанном программно-аппаратном комплексе.

МЕТОДЫ ИССЛЕДОВАНИЯ: экспериментальный метод, сравнение с существующими данными.

ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ: В результате работы была спроектирован и разработан программно-аппаратный комплекс кодирования данных с помощью генетического алгоритма для микроконтроллера Atmega 328.

Содержание

ВВЕДЕНИЕ

В настоящее время в системах управления и обработки данных все чаще применяются микроконтроллеры, решающие широкий спектр задач. Однокристальные микроконтроллеры являются наиболее массовым видом устройств современной микропроцессорной техники, годовой объем выпуска которых, составляет более 2,5 млрд. штук. Интегрируя на одном кристалле высокопроизводительный процессор, память и набор периферийных схем, однокристальные микроконтроллеры позволяют с минимальными затратами реализовать высокоэффективные системы и устройства управления различными объектами (процессами). В отличие от обычных микропроцессоров, для работы которых необходимы внешние интерфейсные схемы, в корпусе однокристальных микроконтроллеров наряду с основными функциональными узлами размещены такие вспомогательные узлы, как тактовый генератор, таймер, контроллер прерываний, цифро-аналоговый и аналого-цифровой преобразователи, порты ввода-вывода.

Согласно отчету исследовательской компании Gartner за 2015 год, объемы продаж 8-ми и 32-битных устройств в долларовом выражении были примерно равны и составляли около $6 млрд. С учетом разницы средних цен, эти цифры говорят о том, что в 2015 году на один встраиваемый 32-битный микроконтроллер приходилось три 8-битных.

Вдобавок к этому 8-ми битные микроконтроллеры продолжают совершенствоваться до сих пор. Например, линейки 8-битных микроконтроллеров PIC и AVR имеют независимую от ядра периферию, работающую без участия центрального процессора и способную обмениваться данными друг с другом. Это помогает повысить эффективность и быстродействие системы при одновременном снижении энергопотребления.

Актуальность использования микроконтроллеров для увеличения быстродействия и совершенствования генетических алгоритмов связана с

возрастающим интересом их применения в автономных системах и развитием нового направления в области проектирования аппаратных средств, получившего название эволюционные аппаратные средства. Как следствие расширение области применения, наиболее остро встает вопрос разработки методов повышения эффективности кодирования данных генетическими алгоритмами. Решением подобных вопросов выступает программная реализация генетических алгоритмов.

На данный момент существует много аналогов программно-аппаратных комплексов, использующих микроконтроллеры со встроенным шифрованием флэш карты. Они используют утилиты для шифрования, такие как TrueCrypt и BitLocker или другие их аналоги. Все они сводятся к одному принципу. Утилита должна быть установлена на флэш карте или на ПК. После подключения к ПК и ввода пароля, флэшка шифрует данные с помощью встроенного микроконтроллера. В следствии чего возникают следующие проблемы:

- необходимость в использовании ПК;
- необходимость в наличии утилиты на ПК;
- совместимость утилит только с определенными операционными системами;
- возможность злоумышленника узнать пароль, так как он будет какое-то время храниться в оперативной памяти ПК;
- долгое завершение работы после шифрования.

Целью данной выпускной квалификационной работы является исключение из процесса шифрования данных на внешнем носителе необходимости применения ПК.

Объект исследования: реализация генетического алгоритма на микроконтроллере Atmega 328.

Предмет исследования: кодирование данных эволюционными методами.

Для достижения обозначенной цели необходимо решить перечень следующих задач:

- провести анализ подходящих микроконтроллеров и методики кодирования данных с помощью генетического алгоритма;
- разработать структурную схему программно-аппаратного комплекса;
- разработать программное обеспечение низкого уровня для микроконтроллера, входящего в состав комплекса;
- исследовать эффективность работы реализованного метода кодирования данных в разработанном программно-аппаратном комплексе.

Для решения этих задач было принято решение разработать программно-аппаратный комплекс кодирования цифровых данных с помощью генетического алгоритма для микроконтроллера Atmega 328.

Пояснительная записка выполнена на 57 страницах без приложения, и 64 с приложением, содержит 20 рисунков, 5 таблиц и приложение.

1 Аналитическая часть

1.1 Применение встроенных контроллеров в технических системах

Программно-аппаратный комплекс - это набор технических и программных средств, работающих совместно для выполнения одной или нескольких сходных задач.

Встроенные контроллеры составляют большую часть производства микропроцессорных средств. Подавляющее большинство современных промышленных, автомобильных, бытовых и других технических систем используют встроенные электронные модули управления, включающие в свой состав управляющий компьютер, называемый микроконтроллером. Термин микро отражает то, что он реализован средствами микроэлектронной технологии в виде чипа, содержащего миллионы транзисторов. Пользователи устройств и систем со встроенными контроллерами могут и не знать, что в них имеется компьютер, используемый для автоматизации процессов управления, обеспечения функциональной гибкости, повышения технического интеллекта системы и качества управления системой. В отличие от обычных персональных компьютеров в микроконтроллерах программа хранится в отдельной энергонезависимой памяти, например флэш-памяти, а в оперативной памяти хранятся только данные, поступающие от датчиков системы, константы и т.п.

С развитием технологии больших интегральных схем встроенные контроллеры стали настолько дешевыми, что их можно найти в большинстве современных устройств и систем. Примеров современных систем, использующих встроенные контроллеры, можно привести очень много, труднее назвать систему, которая не содержит контроллеров.

В таблице 1 приведены некоторые из примеров устройств и систем, позволяющих определить масштабы применений контроллеров.

Таблица 1 - Сферы применений контроллеров

Автомобили	Управление двигателем, антиблокировочные и тормозные системы, управление подушками безопасности, системой микроклимата, навигация gps, системная диагностика.
Бытовая электроника	Телевизоры, посудомоечные машины, плееры DVD, стереосистемы, фотокамеры, часы, автоответчики.
Устройства для компьютера	Клавиатуры, принтеры, сканеры, дисплеи, модемы, жесткие диски.
Электронные компоненты	Осциллографы, вольтметры, генераторы сигналов, логические анализаторы.
Персональные устройства	Сотовые телефоны, переносные плееры MP3, видеоплееры, цифровые камеры.
Авиационные системы	Автопилоты самолетов.

Обобщенная структурная схема контроллерной системы управления техническим объектом управления (ОУ) представлена на рисунке 2. Микроконтроллер (МК) принимает множество информационных сигналов {X} и {U} из дискретных (ДД) и аналоговых датчиков (АД) соответственно из объекта управления (ОУ) и вырабатывает множество управляющих сигналов {Y} в соответствии с алгоритмом управления и выдает их в исполнительные механизмы (ИМ).

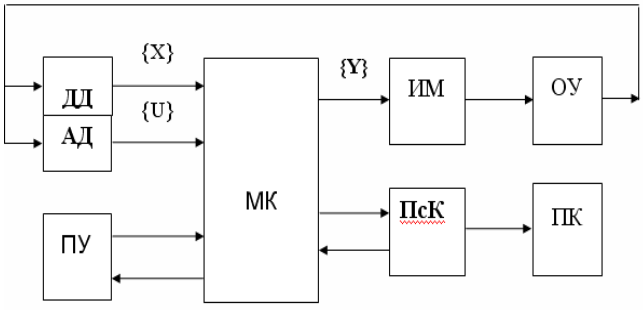

Рисунок 1 – Структурная схема контроллерной системы

Закон управления реализуется контроллером на основе сигналов {X} и {U} из ОУ, алгоритмов их реализации и информации с пульта управления

(ПУ). МК содержит модули, обеспечивающие выполнение программ управления объектом, хранение данных, а также периферийные модули для подключения датчиков и исполнительных механизмов. С помощью ПУ пользователь получает возможность доступа к микроконтроллерной системе: запускать и останавливать её, загружать в контроллер значения некоторых установок (констант), выводить на индикаторы информацию о состоянии объекта и т. п. С помощью последовательного канала связи (ПсК) МК может организовать взаимодействие микроконтроллера с персональным компьютером (ПК) в режиме «запрос – ответ».

В данном пункте была рассмотрена сфера применения и принципы работы микроконтроллеров. Было выяснено, что микроконтроллеры предназначены для управления разными электронными приборами и устройствами. Они используются не только в компьютерах, но и в различной бытовой технике, в роботах на производстве, в телевизорах, в оборонной промышленности. Микроконтроллер является универсальным инструментом, с помощью которого осуществляется управление различной электроникой. При этом алгоритм управляющих команд человек закладывает в них самостоятельно, и может менять его в любое время, в зависимости от ситуации.

1.2 Анализ средств разработки программно-аппаратного комплекса

Программно-аппаратного комплекс на основе микроконтроллера универсален, так как он позволяет повысить технико-экономические показатели, даёт возможность модификации и расширения функциональный возможностей. Перспективным является то, что непрерывное развитие техники приводит к появлению принципиально новых устройств, тестирование которых можно осуществлять разработанным программно-аппаратным комплексом.

Для реализации проекта был выбран микроконтроллер Atmega 328. Он имеет низкое энергопотребление, основанное на усовершенствованной RISC архитектуре.

Atmega 328 - микроконтроллер семейства AVR, имеет 8-битный процессор и позволяет выполнять большинство команд за один такт.

В состав памяти входит:

- 32KB Flash (память программ, имеющая возможность самопрограммирования);
- 2KB ОЗУ;
- 1KB EEPROM (постоянная память данных).

В состав периферийных устройств входит:

- два 8-битных таймера/счетчика с модулями сравнения и делителями частоты;
- 16-битный таймер/счетчик с модулем сравнения и делителем частоты, а также с режимом записи;
- счетчик реального времени с отдельным генератором;
- 6 каналов PWM (аналог ЦАП);
- 6-канальный ЦАП со встроенным датчиком температуры;
- программируемый последовательный порт USART;
- последовательный интерфейс SPI;
- интерфейс I2C;
- программируемый сторожевой таймер с отдельным внутренним генератором;
- внутренняя схема сравнения напряжений;
- блок обработки прерываний и пробуждения при изменении напряжений на выводах микроконтроллера.

К специальным функциям микроконтроллера относят:

- сброс при включении питания и программное распознавание снижения напряжения питания;
- внутренний калибруемый генератор тактовых импульсов;

- обработка внутренних и внешних прерываний;
- 6 режимов сна (пониженное энергопотребление и снижение шумов для более точного преобразования АЦП).

Напряжения питания и скорость процессора:

- 1.8 — 5.5 В при частоте до 4 МГц;
- 2.7 — 5.5 В при частоте до 10 МГц;
- 4.5 — 5.5 В при частоте до 20 МГц.

На рисунке 2 можно увидеть внешнюю часть микроконтроллера.

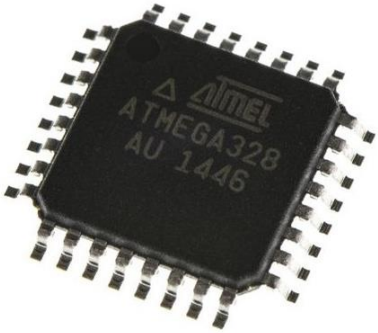

Рисунок 2 – Микроконтроллер Atmega 328

Для работы с большими объемами данных существенно эффективней использовать SD и microSD карты. Они способны хранить информацию считаную с датчиков, регистраторов и преобразователей. Общий объем памяти на платах Arduino составляет до 4 Кбайт. Использование SD карт позволяет увеличить необходимое для любой информации место. Такое решение легкодоступно, не трудоемко и простое в освоении. Одним из способов подключения SD карты к платформе Arduino является использование модуля карты памяти. Он имеет резистор, регулятор напряжения и слот для карты, а так же имеет следующие технические характеристики:

- диапазон рабочих напряжений 4,5-5 В;
- поддержка SD карты до 2 Гб;
- ток 80 мА;
- файловая система FAT 16.

Использование модуля поддерживает чтение, запись и хранение данных н карте памяти. Пример модуля на рисунке 3.

Рисунок 3 – Модуль для чтения карты памяти

У недорогих модулей карт памяти есть и недостатки. Например, самые дешевые и распространенные модели поддерживают только карты до 4Гб и почти все модули позволяют хранить на SD карте файлы объемом до двух гигабайт - это ограничение используемой в большинстве моделей файловой системы FAT.

Для отображения статусов инициализации SD карты, открытия файлов и обработки данных было принято решение использовать LGD дисплей. Существует несколько разновидностей визуализаторов, но самым подходящим оказался дисплей LCD 1602. Он является хорошим выбором для вывода строк символов в различных проектах. Стоит недорого, есть различные модификации с разными цветами подсветки. Недостатком этого экрана является тот факт, что дисплей имеет 16 цифровых выводов, из которых обязательными являются минимум 6. Поэтому использование этого LCD экрана без I2C добавляет серьезные ограничения для платы Arduino Nano. Для экономии контактов дисплей подключался к платформе через интерфейс I2C.

I2C - это протокол, изначально создававшийся для связи интегральных микросхем внутри электронного устройства. Разработка принадлежит фирме Philips. В основе I2C протокола является использование 8-битной шины, которая нужна для связи блоков в управляющей электронике, и системе

адресации, благодаря которой можно общаться по одним и тем же проводам с несколькими устройствами. Данные будут переданы то одному, то другому устройству, добавляя к пакетам данных идентификатор нужного элемента.

Самая простая схема I2C может содержать одно ведущее устройство (чаще всего это микроконтроллер Arduino) и несколько ведомых (например, дисплей LCD). Каждое устройство имеет адрес в диапазоне от 7 до 127. Двух устройств с одинаковым адресом в одной схеме быть не должно.

В работе I2C можно выделить несколько преимуществ:

- для работы требуется всего две линии – SDA (линия данных) и SCL (линия синхронизации);
- подключение большого количества ведущих приборов;
- уменьшение времени разработки;
- для управления всем набором устройств требуется только один микроконтроллер;
- возможное число подключаемых микросхем к одной шине ограничивается только предельной емкостью;
- высокая степень сохранности данных из-за специального фильтра подавляющего всплески, встроенного в схемы;
- простая процедура диагностики возникающих сбоев, быстрая отладка неисправностей;
- шина уже интегрирована в саму Arduino, поэтому не нужно разрабатывать дополнительно шинный интерфейс.

Недостатки:

- существует емкостное ограничение на линии – 400 пФ;
- трудное программирование контроллера I2C, если на шине имеется несколько различных устройств;
- при большом количестве устройств возникает трудности локализации сбоя, если одно из них ошибочно устанавливает состояние низкого уровня.

Таким образом, были рассмотрены основные вопросы использования LCD экрана в сложных проектах Arduino, когда нам нужно экономить свободные контакты на плате. Простой и недорогой переходник I2C позволит подключить LCD экран 1602, занимая два аналоговых контакта. Во многих ситуациях это может быть очень важным.

Также были задействованы часы реального времени на RTC модулях Arduino DS1301. Они используются для генерации случайной последовательности битов используемой эволюционным кодированием данных.

В данном пункте проведен анализ электронных компонентов, были рассмотрены использующиеся в разработанном комплексе периферийные устройства их плюсы и минусы. Также отмечены технические характеристики, назначение и способы подключения к электронной плате. Данные компоненты являются необходимыми для реализации всех функций заложенных в комплекс и имеющие минимальные возможности для простоты работы с ними.

1.3 Сравнение существующих аналогов микроконтроллеров

В настоящее время микроконтроллеры настолько дешевы и доступны, что их обычно используют вместо простых логических схем на основе дискретных компонентов, что позволяет достигнуть гибкости проектирования и сократить площадь, занимаемую на печатной плате. Некоторые машины и роботы сегодня полагаются на огромное количество микроконтроллеров, каждый из которых решает определенную задачу.

Но на рынке сегодня представлено большое количество микроконтроллеров. Что они из себя представляют? И в чем их отличие друг от друга? В этом пункте было проведено сравнение основных семейств микроконтроллеров: AVR, 8051 и PIC.

Микроконтроллер 8051 - это 8-битное семейство микроконтроллеров, разработанное Intel в 1981 году. Это одно из популярных семейств

микроконтроллеров, которые используются во всем мире. Кроме того, этот микроконтроллер изначально назывался «системой на кристалле», поскольку он имеет 128 байт оперативной памяти, 4 Кбайт ПЗУ, 2 таймера, 1 последовательный порт и 4 порта на одном кристалле. Процессор может обрабатывать до 8 бит данных одновременно. Если данные больше 8 бит, то они должны быть разбиты на части, чтобы процессор мог легко их обрабатывать. Большинство микроконтроллеров серии 8051 различных производителей содержат 4 Кбайт ПЗУ, хотя объем ПЗУ может быть расширен до 64 Кбайт. Пример микроконтроллера 8051 на рисунке 4.

Рисунок 4 – Микроконтроллер 8051

Микроконтроллеры 8051 используются в огромном количестве устройств, главным образом потому, что их легко интегрировать в проект. Ниже перечислены основные направления их применения.

Во-первых, это контроль электроэнергии: эффективные системы измерения облегчают контроль использования энергии в домах и производственных помещениях. Эти измерительные системы оптимальны для возможности интеграции микроконтроллеров.

Сенсорные экраны. Большое количество поставщиков микроконтроллеров включает сенсорные функции в свои устройства. Примерами сенсорных экранов на микроконтроллерах являются портативная электроника, такая как сотовые телефоны, медиаплееры и игровые устройства.

Автомобили: микроконтроллеры 8051 находят широкое применение в автомобильных решениях. Они широко используются в гибридных

транспортных средствах для обработки данных с двигателей и управления ими. Кроме того, такие функции, как круиз-контроль и анти-тормозная система, более эффективны с использованием микроконтроллеров.

Медицинские устройства: переносные медицинские устройства, такие как измерители артериального давления и мониторы глюкозы, используют микроконтроллеры для отображения данных, что обеспечивает более высокую надежность при предоставлении медицинских результатов.

Контроллер периферийного интерфейса (PIC) - это серия микроконтроллеров, разработанная компанией Microchip. Микроконтроллер PIC быстрее и проще реализует программы, если сравнивать с другими микроконтроллерами, такими как 8051. Простота программирования и простота взаимодействия с другими периферийными устройствами делает PIC более успешным микроконтроллером.

Пример PIC микроконтроллера на рисунке 5.

Рисунок 5 – Микроконтроллер PIC

PIC - это микроконтроллер, который также состоит из центрального процессора, ОЗУ, ПЗУ, таймеров, счетчиков, АЦП (аналого-цифровых преобразователей) и ЦАП (цифроаналоговых преобразователей). Микроконтроллер PIC также поддерживает протоколы, такие как CAN, SPI, UART для взаимодействия с дополнительными периферийными устройствами. PIC в основном использует модифицированную гарвардскую архитектуру, а также поддерживает RISC (сокращенный набор команд). Благодаря этому PIC

быстрее, чем контроллеры на основе ядра 8051, которые основаны на фон-неймановской архитектуре.

Первые микроконтроллеры AVR были разработаны в 1996 году компанией Atmel (теперь часть Microchip). Проект AVR был разработан Альф-Эгилем Богеном и Вегаром Волланом, поэтому AVR аббревиатура получила две первые буквы от имен разработчиков: Alf-Egil Bogen Vegard Wollan RISC, после эта аббревиатура стала расшифровываться более официально как Advanced Virtual RISC. AT90S8515 был первым микроконтроллером в линейке AVR, хотя первым микроконтроллером, который попал на коммерческий рынок, был AT90S1200 (в 1997 году).

Пример AVR микроконтроллера на рисунке 6.

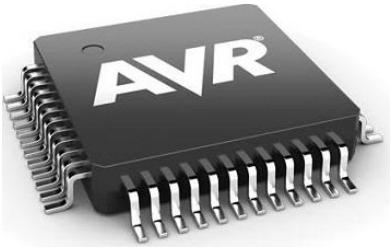

Рисунок 6 – Микроконтроллер AVR

Процессоры AVR широко используются в потребительских электронных устройствах, таких как смартфоны, планшеты, мультимедийные проигрыватели и другие мобильные устройства. Из-за сокращенного набора команд им требуется меньше транзисторов, что позволяет уменьшить размер матрицы интегральной схемы. Процессоры AVR с меньшими размерами уменьшают сложность проектирования и сокращают энергопотребление, что делает их пригодными для более миниатюрных устройств.

Ниже приведена таблица 2 со сравнительными характеристиками рассматриваемых микроконтроллеров.

Таблица 2 – Сравнение микроконтроллеров AVR, 8051 и PIC

	8051	PIC	AVR
Разрядность	8 бит	8/16/32 бит	8/32 бит
Интерфейсы	UART, USART,SPI,I2C	PIC, UART, USART, LIN, CAN, Ethernet	UART, USART, SPI, I2C, иногда CAN, USB, Ethernet
Скорость	12 тактов на инструкцию	4 такта на инструкцию	1 такт на инструкцию
Память	ROM, SRAM, FLASH	SRAM, FLASH	Flash, SRAM, EEPROM
Шинная архитектура	CLSC	Частично RISC	RISC
Архитектура памяти	Фоннеймановская	Гарвардская	Модифицированная
Энергопотребление	75 мА	50 мА	40 мА
Стоимость	150 руб.	240 руб.	102 руб.
Популярные микроконтроллеры	AT89C51, P89v51	PIC18fXX8, PIC16f88X, PIC32MXX	Atmega8, 16, 32; вариации для Arduino

Из таблицы видно, что микроконтроллеры AVR основаны на 8-битных и 32-битных многоядерных процессорах. Процессоры на них предназначены для выполнения меньшего количества инструкций, чтобы они могли работать с большей скоростью, выполняя дополнительные миллионы инструкций в секунду (MIPS). Устраняя ненужные операции и оптимизируя обработку информации, RISC-процессоры обеспечивают большую производительность по сравнению с большинством рассмотренных микроконтроллеров.

Вывод по первому разделу

В данном разделе проведен анализ предметной области, рассмотрены аналоги программных обеспечений и электронных компонентов. На основании выявленных плюсов и минусов было принято решение о создании своего программного обеспечения, так как оно не будет привязано к ПК, а значит, может быть использовано портативным образом. Для аппаратной части был выбран микроконтроллер AVR 328, так как он обладает наибольшим быстродействием из-за меньшей функциональности, низкой энергозатратностью и маленькой стоимостью.

2 Моделирование программно-аппаратного комплекса

2.1 Проектирование генетического алгоритма шифрования данных эволюционным методом

Расширение сферы применения компьютеров выдвигает на первый план решение вопросов необходимого хранения и защиты информации. Объектом защиты информации могут быть сама информация, ее носитель или информационный процесс, в отношении которых необходимо обеспечивать защиту в соответствии с поставленной целью.

Одной из основных характеристик защищаемой информации является ее конфиденциальность, состоящая в известности ее содержания только субъектам, имеющим соответствующие полномочия, как при хранении, так и при передаче по каналам связи. Это обеспечивается скрытностью ее кодирования, называемым шифрованием. Алгоритмы, используемые при шифровании и дешифровании данных, обычно не являются скрытными, а скрытность данных обеспечивается использованием при кодировании специального параметра, называемого ключом, знания которого позволяют правильно расшифровать текст. Для обеспечения полной криптостойкости системы кодирования, ключ, используемый при кодировании должен обладать двумя свойствами:

- должен вырабатываться абсолютно случайным образом и применяться для кодирования только одного текста;
- длина кодируемого текста не должна превышать длину ключа.

Предлагается скрытный код исходного текста получать с помощью генетического алгоритма в процессе эволюционного отбора и без сложных вычислений формулировать секретный код на каждом шаге эволюции.

Основанием к предлагаемому методу кодирования служит исключение ключа при побайтном шифровании с применением многоалфавитной

подстановки в симметричных криптосистемах, когда задаётся последовательность сложения байт открытого текста без явного введения ключа. Подобная операция в генетических алгоритмах называется кроссовером, но, в отличие от побайтного шифрования, пары, над которыми выполняется данная операция, могут выбираться случайным образом (размер блока шифрования величина динамическая), что удовлетворяет одному из требований к криптостойкости шифрования. Второе требование к криптостойкости удовлетворяется, если в процессе эволюции хромосомы (длинны кодов) у потомков последующих поколений будут увеличиваться. Для этого вводятся расширение хромосомы за счет введения дополнительных генов. Выбор длины кода дополнительных генов связывается с допустимыми изменениями кода, что подобно воздействию помех при передаче информации в дискретных технических системах.

Для всех пар родитель-ключ проводят операции кроссовера, которая эквивалентна шифрованию с перестановкой в симметричном кодировании, если ключ расширяет исходное слово, либо к шифрованию с подстановкой, когда части полей исходного кода и кода ключа взаимно заменяются. Это позволяет получить для каждой пары родителей двух потомков, каждый из которых содержит гены обоих родителей. При этом гены первого родителя сосредоточены в одном поле, а второго – в другом. Естественно, в расширение для задания генотипа полученных потомков следует ввести сведения о полях, участвующих в подстановках для каждого потомка. Далее, как и в классическом генетическом алгоритме, полученные коды потомков подвергаются мутации. Перед выполнением мутации необходимо ввести ограничение, связанное с требованиями не вырожденности вида, что можно связать с количеством изменяемых генов в каждом из прародительских полей. В предлагаемом алгоритме это эквивалентно заданию кратности обнаруживаемой и корректируемой ошибки, что может быть обеспечено расширением полей в полученных кодах потомков. Расширение проводится в зависимости от способа обнаружения и коррекции ошибок в сообщениях и

связано с заданием соответствующего преобразования: прямого при кодировании (шифровании) и обратного при дешифровании (восстановлении). Это преобразование также должно присутствовать в поле задания генома. Теперь над расширенными кодами обоих потомков проводится операция мутации, в результате которой формируются новые пары родителей, допускаемых к образованию потомков на следующем шаге генетического отбора. При образовании нового поколения возможен выбор одного из родителей (ключа) из произвольной пары потомков случайным образом, либо использование в каждой из полученных пар в качестве второго родителя (ключа) того из потомков, у которого сохранено большее число неизменных ген первого родителя (ключа). В первом случае вновь имеем дело с общим родителем, и все новые пары образуются с его участием. Это приведет к увеличению количества потомков в последующих поколениях и необходимости введения нового критерия генетического отбора. Во втором случае каждая полученная пара на предыдущем шаге отбора дает одну пару в следующем ряду селекции, что не ведет к увеличению количества потомков по соотношению к первому ряду селекции. Независимо от выбора второго родителя данный подход подобен шифрованию, когда на основе одного ключа генерируется несколько ключей, используемых для скрытного кодирования отдельных сообщений при расширении ключа. Отличие состоит в том, что случайным образом изменяются значения кодируемого сообщения и ключа, что требует введения определенных расширений сообщения и ключа. По критерию минимума ресурсов необходимых для обеспечения скрытности кодирования предпочтение следует отдавать второму случаю.

Для рассмотрения кодирования данных генетическим алгоритмом, его положительных сторон и недостатков, необходимо разобрать этапы преобразований.

Выделим 4 этапа в предлагаемом методе кодирования:

- выбор родителей;
- скрещивание;

- формирование потомка;
- мутация.

Возможно получение любой из существующих симметричных шифров путем изменения последовательностей этапов и критериев отбора.

Алгоритм кодирования и декодирования продемонстрируем на частном случае, где прямо задано поколение, на котором производится селекция, и мутация генов не введена.

Рассмотрим процесс кодирования данных.

Рассматриваем первый и второй биты в закрытом ключе. В зависимости от значений выбираем блок данных для группировки “0” – выбираем первый блок, “1” - выбираем второй блок. Здесь первый бит информации отвечает за первый блок, второй бит за второй блок.

Оставшиеся блоки записываем из первого в третью позицию, из второго в четвёртую. Проводим скрещивание C1 XOR C2 и записываем результат в D1.

Переходим к третьему биту в ключе:

- “0” - берём C1 второй, C2 первый и записываем в D2;
- “1” - берём C1 первый, C2 второй и записываем в D2.

Если количество поколений не удовлетворяет критерию то повторить шаг 1-3, при этом ключ шифрования двигается дальше.

Проводим мутацию путем наложения случайно сгенерированной маски вида 00000100.

Наглядный пример работы алгоритма кодирования:

Ключ шифрования: 011100

Входные блоки: первой блок (01101001), второй блок (10101110), третий блок (11011000), четвертый блок (01001110).

Количество поколений для достижения стойкости: 2

Декодирование данных предполагает наличие секретного ключа использованного при кодировании. Обмен ключом происходит по уже существующим закрытым каналам связи.

На рисунке 7 представлена работа алгоритма данных

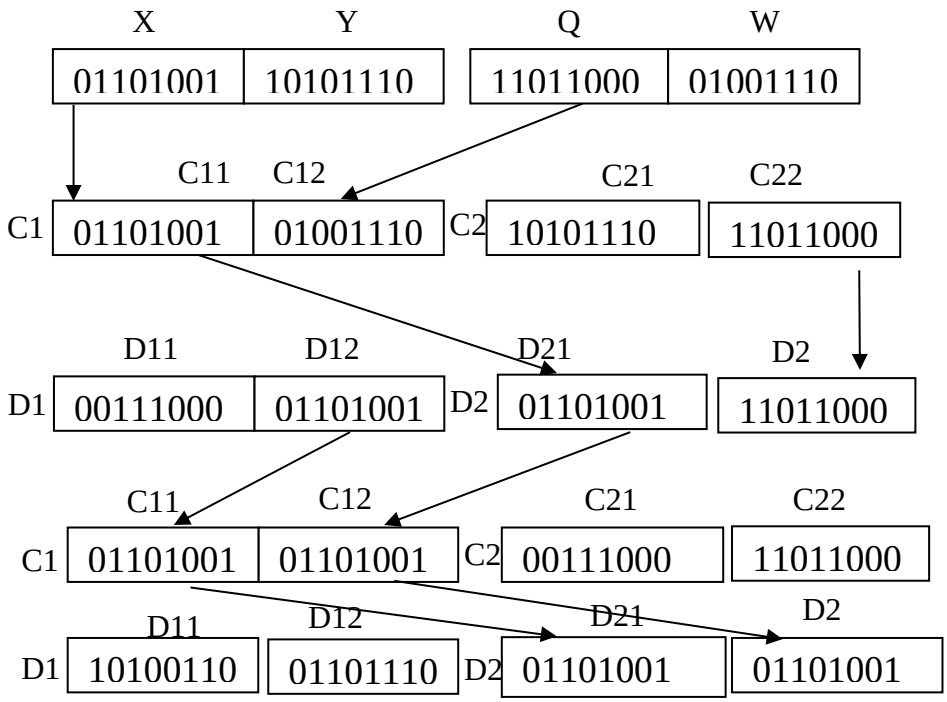

Рисунок 7 - Работа алгоритма кодирования данных

Декодирование:

Сначала идет поиск наложенной маски, затем рассматриваем ключ с конца:

- если бит имеет значение “0”, то C21 в блок D12, а C22 в блок D21;
- если бит имеет значение “1”, то C21 в блок D11, а C22 в блок D22.

В зависимости от значения считанного бита ключа также:

- если “0” то D11=(C11 XOR D21); D22=(C12 XOR D12);
- если “1” то D12=(C12 XOR C22); D21=(C11 XOR C21).

Рассматриваем следующие с хвоста два бита ключа, в зависимости от их значений меняем блоки местами.

Наглядный пример работы алгоритма декодирования данных:

Ключ шифрования: 011100

Входные зашифрованные блоки: первый блок (10100110), второй блок (01101110), третий блок (01101001), четвертый блок (01101001).

На рисунке 8 представлена работа алгоритма декодирования данных.

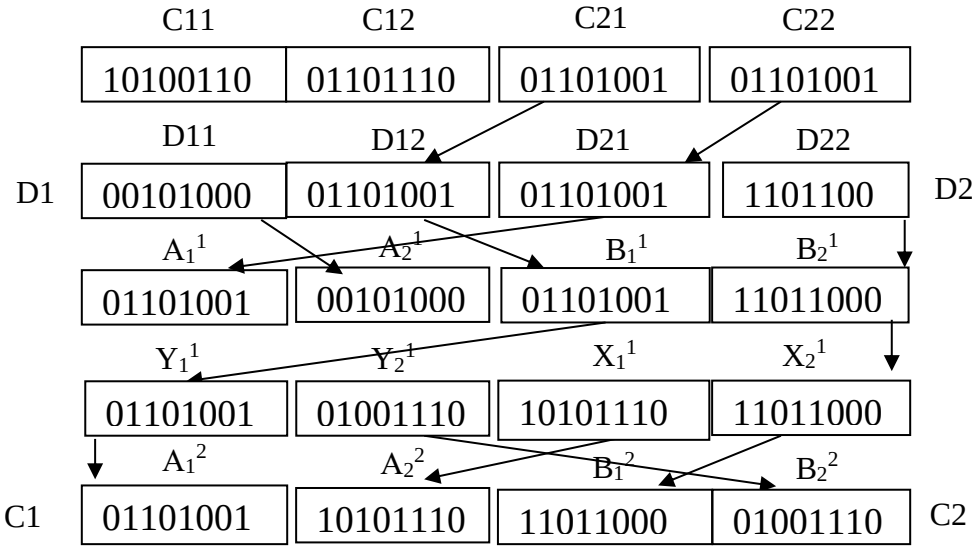

Рисунок 8 - Работа алгоритма декодирования данных

Линейные операции, которые реализуют кодирование и декодирование данных, позволяют поддерживать высокую скорость работы устройства. Крипто стойкость метода шифрования от несанкционированного взлома поддерживается случайно генерируемым закрытым ключом, неизвестным количеством поколений перед селекцией[3].

В данном пункте был рассмотрен генетический алгоритм кодирования данных, в ходе чего выяснено, что использование скрытого кодирования, зависящего от преобразуемых данных, имеет большое теоретическое и практическое значение[4]. Теоретическая значимость заключается в обосновании нового класса примитивов и возможности расширения и совершенствования общих принципов построения итеративных схем блочных алгоритмов.

Теоретически важным является также, то обстоятельство, что математические свойства сложных, на первый взгляд, новых систем эволюционного кодирования данных, связанных с целой системой битовых преобразований, достаточно просто и эффективно определяются аналитически.

2.2 Выбор среды разработки

Для разработки программного обеспечения была выбрана платформа Arduino.

Arduino - это электронный конструктор и удобная платформа быстрой разработки электронных устройств. Платформа пользуется огромной популярностью во всем мире благодаря удобству и простоте языка программирования, а также открытой архитектуре и программному коду. Устройство программируется через USB без использования программаторов.

Arduino позволяет компьютеру выйти за рамки виртуального мира и взаимодействовать с ним. Устройства на базе Arduino могут получать информацию об окружающей среде посредством различных датчиков, а также могут управлять различными исполнительными устройствами.

Микроконтроллер на плате программируется при помощи языка Arduino (основан на языке Wiring) и среды разработки Arduino (основана на среде Processing). Проекты устройств, основанные на Arduino, могут работать самостоятельно, либо же взаимодействовать с программным обеспечением на компьютере (напр.: Flash, Processing, MaxMSP). Платы могут быть собраны пользователем самостоятельно или куплены в сборе. Программное обеспечение доступно для бесплатного скачивания. Исходные чертежи схем (файлы CAD) являются общедоступными, пользователи могут применять их по своему усмотрению.

Программа для Arduino пишется на C++, дополненным простыми и понятными функциями для управления вводом/выводом на контактах. Для удобства работы с Arduino существует бесплатная официальная среда программирования «Arduino IDE», работающая под Windows, Mac OS и Linux. С помощью неё загрузка новой программы в Arduino становится мгновенной, стоит лишь подключить плату к компьютеру через USB. Хотя возможна разработка и через Visual Studio, Eclipse, другие IDE или командную строку.

Ещё одной отличительной особенностью Arduino является наличие плат расширения, так называемых shields или просто «шилдов». Это дополнительные платы, которые ставятся поверх Arduino, добавляя ему новые возможности. Так, например, существуют платы расширения для подключения к локальной сети и интернету (Ethernet Shield), для управления мощными моторами (Motor Shield), для получения координат и времени со спутников GPS (модуль GPS) и многие другие.

Для разработки аппаратного обеспечения была выбрана плата Arduino Nano.

Arduino Nano – это небольшая, полнофункциональная отладочная плата, адаптированная для работы с макетными платами, построенная на базе микроконтроллера ATmega328 (Arduino Nano 3.x) или Atmega168 (Arduino Nano 2.x). Она обладает той же функциональностью, что и Arduino Duemilanove, но имеет меньшие размеры. Она отличается только отсутствием разъема питания и работой через mini-USB. Arduino Nano разработана и производится компанией Gravitech. На рисунке 9 представлен внешний вид платы Arduino Nano.

Рисунок 9 - Arduino Nano

Она имеет следующие технические характеристики:

- микроконтроллер Atmega 168 или Atmega 328;
- рабочее напряжение (логический уровень) 5В;
- входное напряжение (рекомендуемое) 7-12В;
- входное напряжение (предельное) 6-20В;
- 14 цифровых выводов типа ввод/вывод, 6 из которых можно использовать в качестве ШИМ выходов;

- 8 аналоговых входных вывода;
- постоянный ток через входные/выходные выводы 40 мА;
- флеш-память 16 Кб (ATmega168) или 32 Кб (ATmega328), из которых 2 Кб используются загрузчиком;
- оперативная память SRAM 512 б (ATmega168) или 1 Кб (ATmega328);
- тактовая частота 16 МГц.

Arduino Nano может питаться через mini-B USB соединение, от внешнего нестабилизированного источника питания 6–20 В (вывод 30) или от стабилизированного источника напряжения 5В (вывод 27). Источник питания с наибольшим напряжением выбирается автоматически.

Для сравнения был выбран одноплатный компьютер Raspberry Pi. Он обладает всеми атрибутами настоящего компьютера: выделенным процессором, памятью и графическим драйвером для вывода через HDMI. На нем может работать специальная версия операционной системы Linux. Поэтому на Raspberry Pi легко установить большинство программ для Linux. Хотя в Pi и отсутствует внутреннее хранилище данных, на этом компьютере можно использовать смарт-карты в качестве флэш-памяти, обслуживающей всю систему. Требования к электропитанию для этих двух систем очень отличаются. Raspberry Pi для работы нужно постоянное напряжение 5V, более того, работа Raspberry Pi завершается программным процессом как у обычного компьютера. Raspberry Pi сложно переносить с места на место, так как нельзя просто вставить в него две батарейки AA. Для работы этого компьютера необходимо обеспечить бесперебойное питание, а также подключить дополнительное оборудование, которое гарантирует подачу постоянного тока.

Для сравнения с микроконтроллерами семейства Arduino был выбран Arduino Uno. Количество digital контактов совпадает с Nano, а analog контактов и того меньше (8 у Nano и 6 у Uno). Тактовая частота зависит от модели процессора. С учетом выбора Atmega 328, которая используется в

обеих платах, разницы не будет. Рабочее напряжение одинаковое, а ОЗУ меньше в Nano (2 Кб в Uno и 1 Кб в Nano). С учетом, что в оперативной памяти будут храниться лишь блоки данных для преобразования, которые не будут превышать её размер, преимуществ от этого тоже не будет. В завершении Nano меньше по своим габаритам, чем Uno, что по итогу со всеми другими качествами делает выбор в пользу Nano оправданным.

В таблице 3 приведено сравнение микропроцессора Arduino и одноплатным компьютером Raspberry Pi.

Таблица 3 - Сравнение плат

	Arduino Nano	Arduino Uno	Raspberry Pi
Производительность	1 ядро 16 МГц 1 Кб ОП	1 ядра 16 МГц 1 Кб ОП	4 ядра 1200 МГц 1 Гб ОП
Длительность работы от батареек	Потребляет 40 мА	Потребляет 40 мА	Потребляет 2000 мА
Скорость реакции в проектах критичных к времени	100% контроль над временем и длительностью подачи сигналов	100% контроль над временем и длительностью подачи сигналов	Из-за многозадачности критический процесс может проспать своё время
Стоимость	От 100 рублей	От 200 рублей	От 2000 рублей

Исходя из вышеперечисленного, целесообразность использования Arduino Nano оправдана своими минимальными возможностями, низкой энергозатратностью и высокой быстродействественностью. Её мощности недостаточно для того, чтобы собрать какое-либо сложное устройство, но вполне достаточно для различных простейших систем. Это плата довольно компактная, удобная и обладает всеми возможностями Arduino Uno.

2.3 Проектирование и сборка аппаратного обеспечения устройства кодирования и декодирования данных

В связи с тем, что плата Arduino Nano содержит два контакта GND и один 5V, а имеется три периферийных устройства и кнопка, было принято решение собирать аппаратное обеспечение на макетной плате размером 10x17 контактов. Размеры были выбраны исходя из минимально необходимых условий, дабы не нарушать компактность комплекса.

После вставки платы в макетку для каждого контакта Arduino появляются еще 3 взаимосвязанные позиции.

Для подключения модуля карты памяти используется 6 контактов, взаимодействие производится по интерфейсу SPI. Она выглядит на плате как разъем на лицевой поверхности с шестью штырями. Чтобы подключить карту, нужны сам контроллер, модуль карты и 6 проводов. Помимо SPI существует режим SDIO, но он сложен в реализации и слабо совместим с Arduino. SPI легко налаживается для работы со всеми микроконтроллерами, поэтому использовался он.

Подключение цифровых выводов производится следующим образом:

- MOSI подключается к D11;
- MISO к D12;
- SCK к D13;
- CS к 4;
- VCC на +5В;
- GND к GND.

На плате имеются разъемы для подключения к 3,3 и 5 вольтам. Питание самой карты составляет 3,3 вольт, поэтому проще применять микроконтроллер с таким же питанием, в ином случае нужен преобразователей уровней напряжения. На самых распространенных платах Arduino такой выход есть.

Для удобства работы с внешними накопителями данных в среде Arduino IDE доступны уже готовые библиотеки. Ничего дополнительно скачивать или устанавливать в большинстве случаев не понадобится.

Для подключения библиотеки в скетче нужно использовать инструкцию include:

#include <SPI.h> и #include <SD.h>

Библиотека SPI нужна для правильной работы устройств, подключаемых по SPI. Библиотечные функции нужно для считывания и записи данных на карту. Библиотека может поддерживать SD и SDHC карты. Имена записываются в формате 8.3, то есть 8 знаков для названия, 3 для расширения. Путь к файлу записывается с помощью слэшей «/». Ардуино-библиотека SD содержит различные функции, с помощью которыми можно управлять данными. Функции класса SD:

- begin() – функция инициализирует библиотеку, присваивает контакт для сигнала;
- exists() – призвана проверить, имеется ли на карте необходимая информация;
- mkdir() – позволяет создать нужную папку на карте памяти;
- rmdir() – с помощью этой функции можно удалить папку. Важно, чтобы удаляемая папка была пустой;
- open() – позволяет открыть файл, который нужен для записи или чтения. Если нужный файл отсутствует на карте, то он будет создан;
- remove() – удаляет любой файл.

В ответ на все эти функции должно прийти одно из значений – true, в случае удачного завершения операции и false при неудаче.

Схема подключения модуля карты памяти к Arduino Nano представлена на рисунке 10. Она была разработана с помощью инструмента “Circuito.io”.

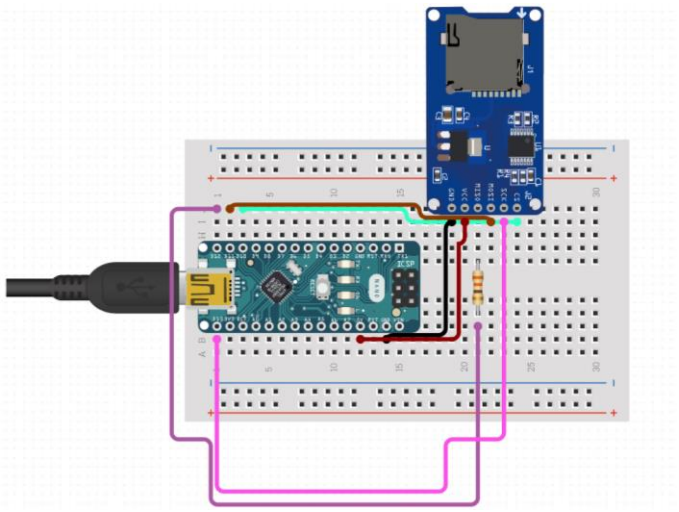

Рисунок 10 – Схема подключения модуля карты памяти

Далее подключаем LCD 1602 дисплей. Жидкокристаллический монитор с поддержкой i2c подключается к плате при помощи четырех проводов – два провода для данных, два провода для питания.

- вывод GND подключается к GND на плате;
- вывод VCC – на 5V;
- SCL подключается к пину A5;
- SDA подключается к пину A4.

Для взаимодействия Arduino с LCD 1602 по шине I2C вам потребуются как минимум две библиотеки:

- Wire.h;
- LiquidCrystal_I2C.h.

Она включает в себя большое разнообразие команд для управления монитором по шине I2C и позволяет сделать скетч проще и короче. Нужно дополнительно установить библиотеку. После подключения дисплея нужно дополнительно установить библиотеку LiquidCrystal_I2C.h.

Описание функций и методов библиотеки LiquidCrystal_I2C:

- home() и clear() – первая функция позволяет вернуть курсор в начало экрана, вторая тоже, но при этом удаляет все, что было на мониторе до этого;

- write(ch) – позволяет вывести одиночный символ ch на экран;
- cursor() и noCursor() – показывает/скрывает курсор на экране;
- blink() и noBlink() – курсор мигает/не мигает (если до этого было включено его отображение);
- display() и noDisplay() – позволяет подключить/отключить дисплей;
- scrollDisplayLeft() и scrollDisplayRight() – прокручивает экран на один знак влево/вправо;
- autoscroll() и noAutoscroll() – позволяет включить/выключить режим авто прокручивания;
- leftToRight() и rightToLeft() – Установка направление выводимого текста – слева направо или справа налево;
- createChar(ch, bitmap) – создает символ с кодом ch (0 – 7), используя массив битовых масок bitmap для создания черных и белых точек;

Схема подключения LCD дисплея и модуля карты памяти к Arduino Nano представлена на рисунке 11.

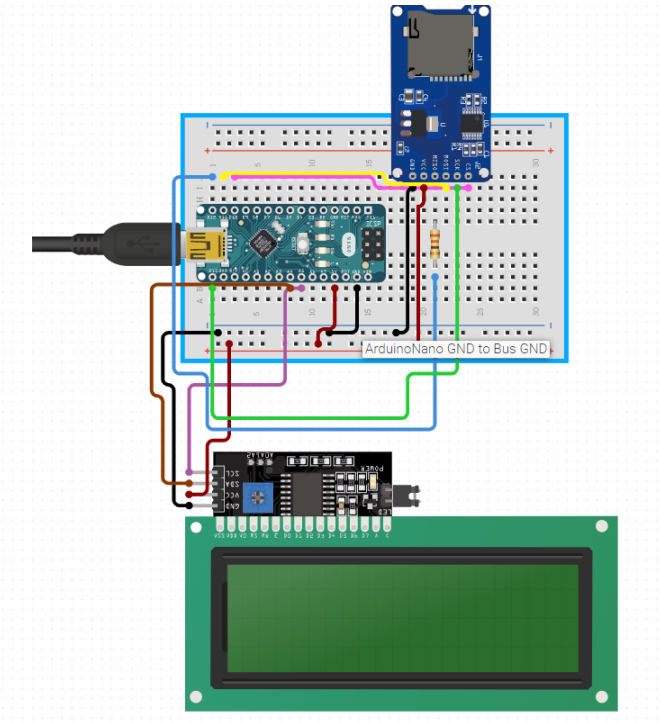

Рисунок 11 – Схема подключения модуля карты памяти и LCD 1602

Затем подключаем RTC часы DS3231. Чтобы это сделать нужно:

- VCC подключается к 5В на Ардуино;
- GND на RTC – к земле на Ардуино;
- SDA – A4;
- SCL – A5.

Для начала работы с модулем часов нужно установить библиотеки DS1307RTC, TimeLib и Wire. Можно использовать для работы и RTCLib.

#include <iarduino_RTC.h> - подключаем библиотеку.

Функция begin() - инициализация работы RTC модуля.

Функция settime **(**сек, мин, час, день, мес., год, дн.**)** - установка времени;

Функция gettime **(**строка**)** - чтение времени;

функция blinktime **(**частота**)** - заставляет функцию gettime «мигать» указанным параметром времени;

функция period **(**минуты**)** - указывает минимальный период обращения к модулю в минутах;

Переменная seconds - возвращает секунды от 0 до 59;

Переменная minutes - возвращает минуты от 0 до 59;

Переменная hours - возвращает часы от 1 до 12;

Переменная Hours - возвращает часы от 0 до 23;

Переменная midday - возвращает полдень 0 или 1 (0-am, 1-pm);

Переменная day - возвращает день месяца от 1 до 31;

Переменная weekday - возвращает день недели от 0 до 6 (0-воскресенье, 6-суббота);

Переменная month - возвращает месяц от 1 до 12;

Переменная year - возвращает год от 0 до 99.

Схема подключения RTC часов, LCD дисплея и модуля карты памяти к Arduino Nano представлена на рисунке 12.

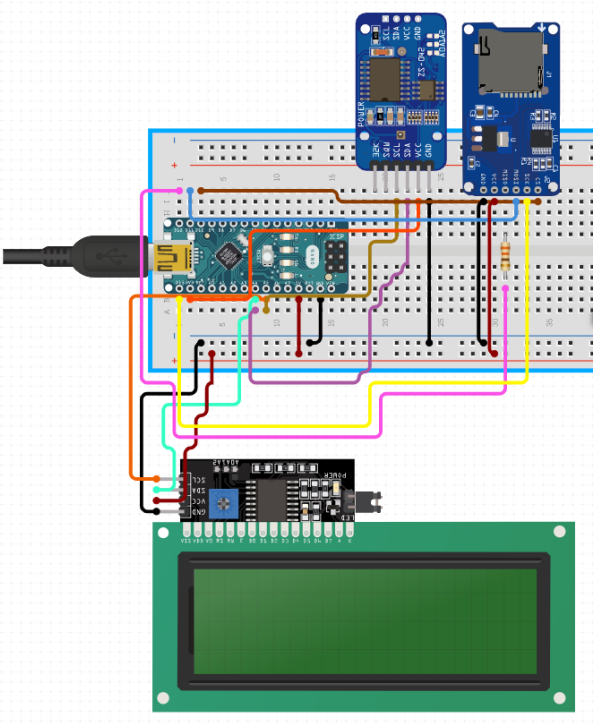

Рисунок 12 – Схема подключения RTC часов модуля карты памяти и LCD 1602

В завершении сборки, для автономности комплекса был использован самодельный блок питания. В его состав входит:

- DC-DC преобразователь;

Входное напряжение 2,5В (мин) – 5,5В (макс). Выходное напряжение 5В, может быть 1,5А на большие расстояния. Частота переключения 1 МГц. Статичное потребление энергии 80 uA. Размер доски 4,2*1,5*5 см. На рисунке 13 представлен внешний вид DC-DC преобразователя.

Рисунок 13 – DC-DC преобразователь

- TP4056 5В Micro USB 1А, зарядное устройство (модуль переразряда);

Входное напряжение 5В. Напряжение зарядки 4.2В. Максимальный ток зарядки 1000мА. Избыточное напряжение защиты 2,5В. Текущий ток защиты 3А. Размер доски 2,6*1,7 см. На рисунке 14 представлен внешний вид зарядного устройства TP4056.

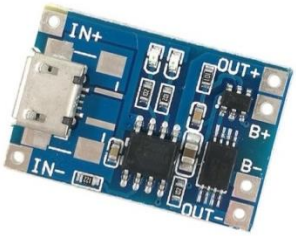

Рисунок 14 – TP4056 зарядное устройство

- батарея ноутбука Acer Iconia A3.

Батарейная ячейка 4-клетка. Ёмкость 27WH. Напряжение 3.7 В.

На рисунке 15 представлена батарея ноутбука Acer Iconia A3.

Рисунок 15 – Батарея ноутбука Acer Iconia A3

В данном пункте выпускной квалификационной работы было спроектировано и собрано аппаратное обеспечение кодирования и декодирования данных. Была спроектирована структурная схема подключения всех периферийных устройств к плате, обозначены используемые библиотеки для работы с ней, а также доступные функции в них.

2.4 Проектирования программного обеспечения

2.4.1 Построение блок-схемы

Одним из наиболее трудоемких этапов решения задачи на ЭВМ является разработка алгоритма.

Под алгоритмом понимается точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату.

Алгоритмы решения практических задач обычно имеют комбинированную структуру, то есть включают в себя все три типа вычислительных процессов.

К изобразительным средствам описания алгоритмов относятся следующие основные способы их представления:

- словесный (записи на естественном языке);
- структурно-стилизованный (записи на алгоритмическом языке и псевдокод);
- графический (изображение схем и графических символов);
- программный (тексты на языках программирования).

Графический способ описания алгоритма предполагает, что для описания структуры алгоритма используется совокупность графических изображений (блоков), соединяемых линиями передачи управления. Такое изображение называется методом блок-схем.

Блок-схема алгоритма - это графическое представление хода решения задачи. Блок-схема состоит из блоков, соединенных линиями, а блоки изображаются в виде геометрических фигур, называемых символами. Внутри символов записываются указания о выполняемых блоком функциях – формулы, текст, логические выражения. Вид символов и правила выполнения блок-схем стандартизированы – ГОСТ 19.701-90 содержит перечень символов, их наименования, отображаемые функции, формы и размеры, а также правила

выполнения схем. При разработке алгоритма каждое действие обозначается соответствующим блоком, показывая их последовательность линиями со стрелками на конце.

Для написания программного обеспечения была создана блок-схема алгоритма. На рисунке 16 приведена общая блок-схема работы программы.

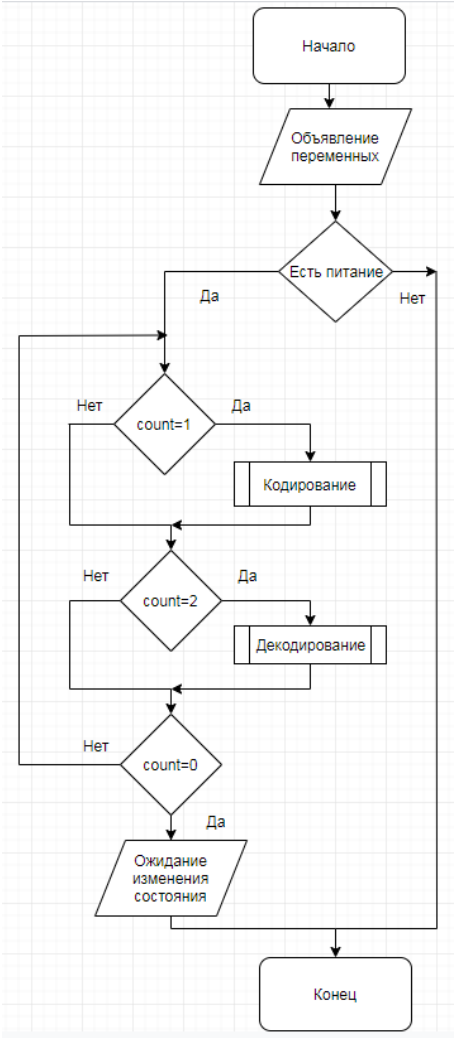

Рисунок 16 – Общая блок-схема алгоритма

Данная блок-схема показывает, что программа объявляет статичные переменные для дальнейшей работы при первом запуске и ждет, пока не появится питание на плате. После появления питания программа начинает в непрерывном режиме, реализуемом функцией void loop(), сравнивать счетчик count. В данном случае, если счетчик равен нулю, то программа ждет событие,

которое изменит счетчик. Изменить счетчик можно нажатием одной из двух копок. Она, изменит счетчик на цифру “1”, другая на “2”. Кодирование файла произойдет в случае, если счетчик будет равен единице, декодирование, если двум. В соответствующую функцию будут переданы переменные, содержащие имена файлов для чтения исходного файла и записи конечных данных в другой файл. После выполнения функции счетчик переходит в режим ожидания изменения и принимает значение равное нулю.

Далее рассмотрены функции кодирования и декодирования данных. На рисунке 17 представлена блок-схема алгоритма кодирования данных.

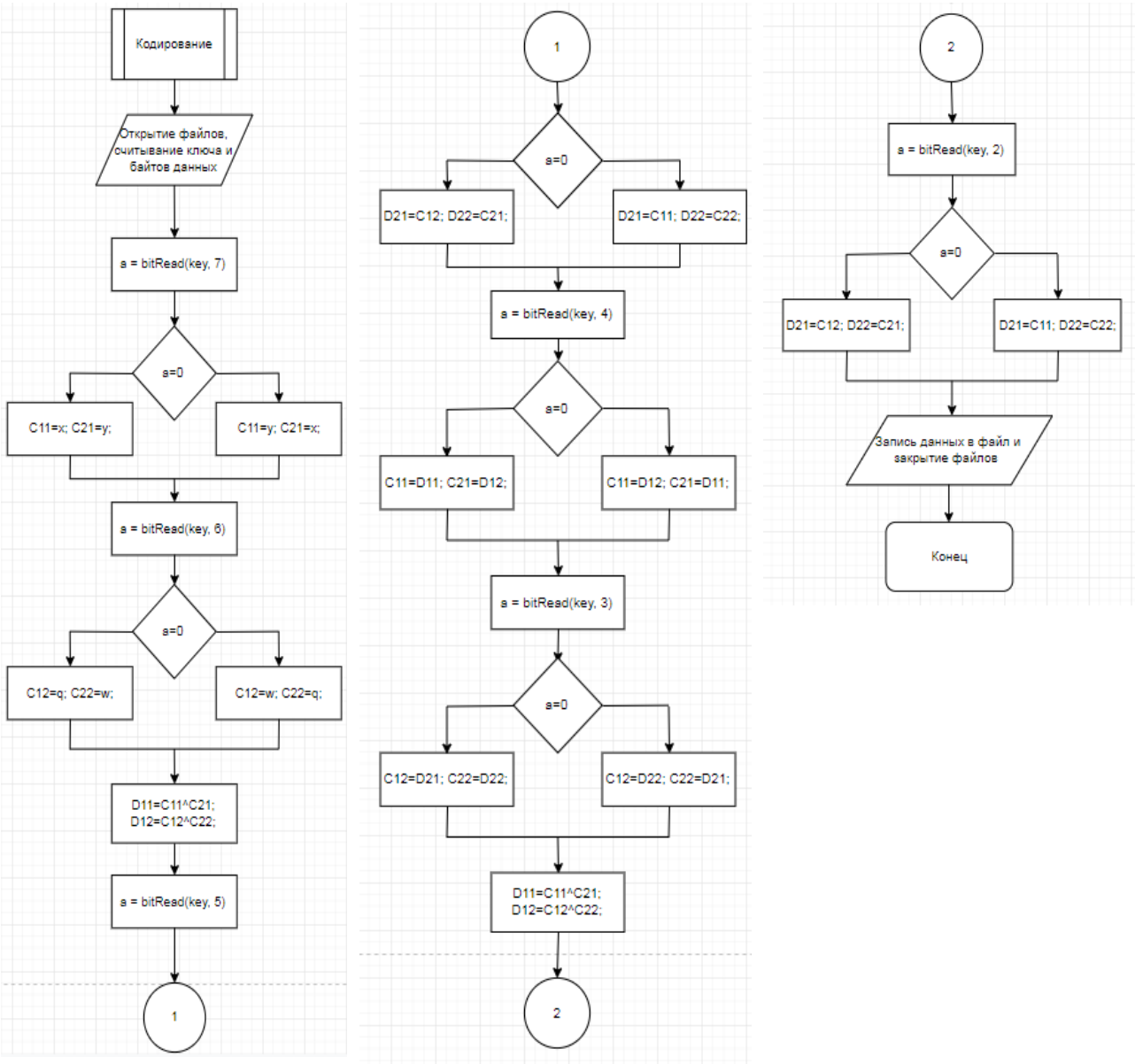

Рисунок 17 – Блок-схема алгоритма кодирования данных

После получения функцией имён файлов на открытие, она их открывает и считывает ключ шифрования с ещё одного файла. В переменные записывается группа данных для кодирования, после чего функция начинать считывать ключ, побитого начиная со старшего бита. Следуя методу шифрования, в зависимости от значений ключа, функция преобразовывает данные. После этого она записывает готовую информацию в файл для записи и считывает следующую группу данных. Когда все данные будут зашифрованы, функция закроет всё открытые файлы, выведет сообщение на дисплей о том, что процесс прошёл успешно и переведет счетчик в состояние ожидания.

Когда кнопка декодирования будет нажата, счетчик примет значение “2” и запустится следующая функция. На рисунке 18 представлена блок-схема алгоритма декодирования данных

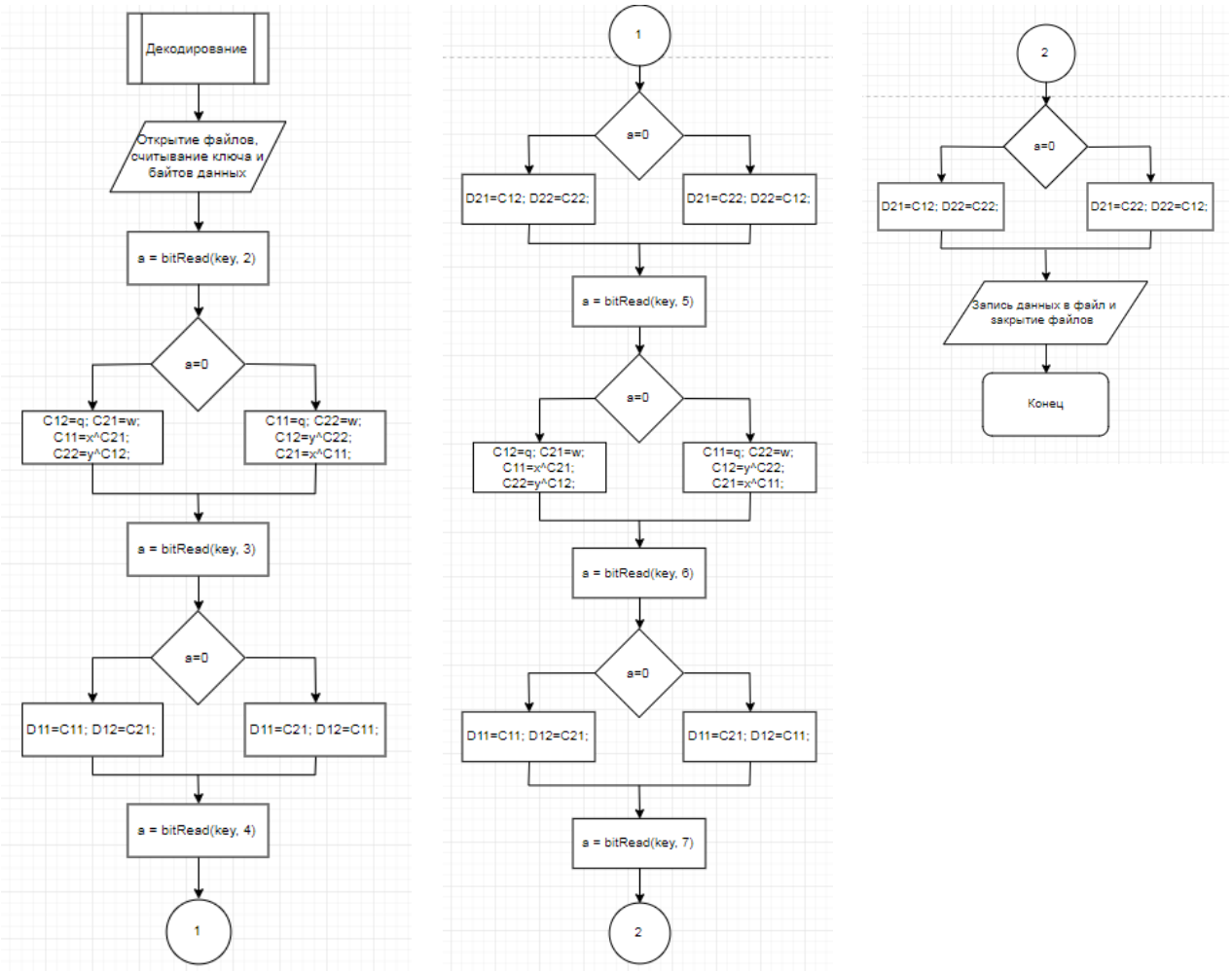

Рисунок 18 – Блок-схема алгоритма декодирования данных

Данная функция принимает имя файла с закодированными данными и имя файла, куда запишет расшифрованную информацию. Также в ней открывается и считывается ключ шифрования, который был сгенерирован и записан в файл в предыдущей функции. Считывая ключ в обратном порядке, то есть с последнего используемого бита, функция генерирует исходные данные по их преобразованному виду, следуя алгоритму декодирования.

В данном пункте был рассмотрен алгоритм работы программного обеспечения, на основании которого можно сделать вывод о том, что данный алгоритм имеет однопоточную структуру, задействует минимальное количество разнообразных операций и оптимизирован под экономию энергии. Всё это полностью реализуется использованием выбранной аппаратуры.

2.4.2 Анализ программного кода

В ходе выполнения дипломной работы был написан программный код на языке C++ который составляет 220 строк, реализующий взаимодействие со сторонними устройствами, вместе образующий комплекс кодирования данных. Разработка программы велась в среде программирования Arduino IDE.

```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <iarduino_RTC.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2)
```

Для начала, подключаются библиотеки для работы платы с LCD дисплеем, модулем карты памяти и часами реального времени. Также инициализируется модель дисплея:

```
LiquidCrystal_I2C lcd(0x27,16,2)
```

Затем объявляются переменные типа File для взаимодействия с файлами расположенными на карте памяти. Одна переменная отвечает за файл для

обработки информации, и он открывается на чтение, вторая для записи преобразованной информации, он открывается на запись. Третья переменная открывает файл с ключом. В зависимости от того какая будет выполняться функция файл откроется на запись или чтение.

```
File dataFile1
File dataFile
File FileKey
```

После этого объявляется переменная для открытие последовательного порта и инициализации SD карты.

```
const int chipSelect = 4
```

Объявляется переменная типа char для конвертирования зашифрованного блока данных в вид, в котором он будет храниться в файле.

```
char buff
```

Следующий блок программного кода объявляет переменные связанные непосредственно с алгоритмом шифрования: значения считаных блоков с файла, ключ, счетчик выбора функции, блоки банных для преобразования информации.

```
int x,y,q,w
int key
int count=0
int a=0
int b=0
int d=0
int C11, C12, C21, C22, D11, D12, D21, D22
```

Последний блок кода объявления переменных нужен для хранения имен файлов и работы с ними.

```
String gen1
String gen2
String str = "EXAMPLE.txt"
String str1 = "EXAMPL1.txt"
```

String str2 = "EXAMPL2.txt"

Глобальная функция void setup() служит местом, где могут быть выполнены команды непосредственно в момент загрузки программы. Они выполнятся только один раз при старте системы. Эта функция ничего не возвращает, так как она имеет тип void. Все желаемые команды располагаются в фигурных скобках после неё.

Далее объявляется работа последовательного порта и скорость отображения данных. Он нужен для проверки хода работы программы в режиме отладки с помощью ПК.

Serial.begin(9600)

Затем открываются digital порты D3 и D4 на вход, и подается высокий уровень на них для определения текущего состояния кнопки. При нажатии на неё считывается сопротивление и в зависимости от кнопки, будет выполняться кодирование или декодирования данных.

pinMode(2, INPUT)

pinMode(3, INPUT)

digitalWrite(2, HIGH)

digitalWrite(3, HIGH)

Функция while (!Serial) служит для ожидания подключения последовательно порта. Для визуального определения момента выполнения этой части кода был добавлен вывод соответствующего сообщения на дисплей

lcd.init() //инициализация дисплея, пишется один раз

lcd.backlight() //включение подсветки, пишется один раз

lcd.clear() //очистка дисплея, указывается при каждом использовании

lcd.setCursor(0,0) //установка курсора в первый ряд на первую позицию

lcd.print("Initializing") //сообщение, выведенное в первой строке

lcd.setCursor(0,0) //установка курсора во второй ряд на первую позицию

lcd.print("SD card...") //сообщение, выведенное во второй строке

Условие if (!SD.begin(chipSelect)) определяет библиотеку SD и карту. chipSelect - контакт, подключенный к линии выбора микросхемы SD-карты.

Если подключиться не удалось, выполнит следующие команды:

```
lcd.clear()
lcd.setCursor(0,0)
lcd.print("Card failed")
lcd.setCursor(0,1)
lcd.print("or not present")
```

Если всё прошло успешно, то программа выполнит:

```
lcd.clear()
lcd.setCursor(0,0)
lcd.print("Card")
lcd.setCursor(0,1)
lcd.print("initialized")
```

На этом, действие функции void setup() заканчивается и начинает выполнять void loop().

Функция void loop – место, где все содержащиеся в ней команды будут выполняться бесконечно, пока на плате будет подано электричество. Это главная функция, которая является точкой входа в программу.

На этом этапе открывается сеанс работы RTC часов.

```
RTC.begin()
```

Далее считывается сопротивление с кнопки, тем самым определяется её использование.

```
int but2 = digitalRead(2)
int but3 = digitalRead(3)
```

После этого оператор if проверяет полученные показатели, на основании которых присваивает переменным открывающим файлы для работы соответствующие имена. В зависимости от выполненного условия выводится сообщение о виде преобразования данных и переводится счетчик в режим ожидания принимая значение равное “0”.

```
if((but2 == HIGH)&&(b == 1)){
b = 0;}
```

```
if((but2 == LOW)&&(b == 0)){
b = 1;
gen1 = str;
gen2 = str1;
count = 1;
lcd.clear(); lcd.setCursor(0,0); lcd.print("crypt");}
if((but3 == HIGH)&&(d == 1)){
d = 0;}
if((but3 == LOW)&&(d == 0)){
d = 1;
gen1 = str1;
gen2 = str2;
count = 2;
lcd.clear(); lcd.setCursor(0,0); lcd.print("decrypt");}
```

Как только счетчик изменит свое значение с нуля на “1” или “2” один из двух операторов if начнет выполнять команды из своего блока. Далее рассмотрен блок кода, отвечающий за кодирование информации.

После выполнения условия входа программа считывает значение текущих секунд на RTC часах и на основе этих данных с помощью оператора random, генерирует случайный ключ. Секунды в данном случае являются нижним порогом для генерируемого значения ключа. Затем он записывается в отдельный файл.

```
DateTime now = RTC.now()
randomSeed(now.second(), DEC)
key = randomSeed
FileKey = SD.open("Key.txt")
if(FileKey){buff = (char) key; FileKey.print(buff);}
FileKey.close()
```

Далее открывается два файла, с которыми программа будет работать.

```
dataFile = SD.open(gen1)
```

dataFile1 = SD.open(gen2, FILE_WRITE)

Из файла считывается первые 4 байта и разбиваются на блоки по 8 бит.

x = dataFile.read()

y = dataFile.read()

q = dataFile.read()

w = dataFile.read()

Берется старший бит в блоке ключа:

a = bitRead(key, 7)

и блоки перемешиваются в зависимости от значения бита.

if(a==0) {C11=x; C21=y;}

else {C11=y; C21=x;}

a = bitRead(key, 6);

if(a==0) {C12=q; C22=w;}

else {C12=w; C22=q;}

После этого в новые блоки данных формируются путем проведения операции:

D11=C11^C21

D12=C12^C22

И снова берётся бит из ключа и идет формирование последний двух блоков данных.

a = bitRead(key, 5);

if(a==0) {D21=C12; D22=C21;}

else {D21=C11; D22=C22;}

Эта процедура повторяется ещё раз.

a = bitRead(key, 4);

if(a==0) {C11=D11; C21=D12;}

else {C11=D12; C21=D11;}

a = bitRead(key, 3);

if(a==0) {C12=D21; C22=D22;}

else {C12=D22; C22=D21;}

```
D11=C11^C21;
D12=C12^C22;
a = bitRead(key, 2);
if(a==0) {D21=C12; D22=C21;}
else {D21=C11; D22=C22;}
```

Последней операцией преобразования данных в блоке является выполнение функции мутации, в закодированном сообщении, по средствам наложения маски сформированной генератором случайных чисел.

Далее в файл для записи помещаются преобразованные блоки данных. Процедура повторяется до тех пор, пока в файле для чтения есть данные.

```
buff = (char) D11
dataFile1.print(buff)
buff = (char) D12
dataFile1.print(buff)
buff = (char) D21
dataFile1.print(buff)
buff = (char) D22
dataFile1.print(buff)
```

По завершения всей процедуры все файлы закрываются, на дисплей выводится сообщение, что всё прошло успешно и счетчик переводится в состояние ожидания.

```
dataFile.close()
dataFile1.close()
lcd.clear(); lcd.setCursor(0,0); lcd.print("Successfully")
count = 0
```

При выполнении функции декодирования данных, она открывает файл с ключом и, считывая биты начиная с младшего, расшифровывает блоки данных, используя алгоритм в обратном порядке. Полный код программы представлен в приложении А.

Вывод по второму разделу

В данной главе были рассмотрены принципы работы генетического алгоритма на примере кодирования байтовых блоков данных. Также выбрана среда разработки и плата, к которой будут подключаться всё периферийные устройства. Было проведено сравнение с аналогом платы семейства Arduino и одноплатным компьютером Raspberry Pi. Построена блок-схема работы алгоритма программы и в ходе её рассмотрения выявлены преимущества в использовании Arduino Nano. Рассмотрен программный код системы и разобраны используемые функции.

3. Проектная часть

3.1 Тестирование программно-аппаратного комплекса

Тестирование программно-аппаратного комплекса - проверка соответствия между реальным и ожидаемым поведением комплекса, осуществляемая на конечном наборе тестов, выбранном определенным образом.

Цели тестирования:

- повысить вероятность того, что приложение, предназначенное для тестирования, будет соответствовать всем описанным требованиям;
- предоставление актуальной информации о состоянии продукта на данный момент.

В тестовой части было выполнено преобразование данных в обе стороны, используя текстовые файлы размером от 1 Кб до 10 Кб включительно. Целью тестирования было определить зависимость времени преобразования файла от его размера. Результаты представлены в таблице 4.

Таблица 4 - Шифрование с помощью микроконтроллера

Размер файла в Кб	Кодирование в секундах	Декодирование в секундах
1	20,718	20,888
2	41,353	41,456
3	59,304	59,421
4	78,977	79,083
5	98,8	98,935
6	118,568	118,651
7	138,258	138,359
8	157,96	158,073
9	177,709	177,835
10	206,547	206,801

Из таблицы следует, что скорость шифрования данных составляет в среднем 49,6 байт в секунду.

Процесс декодирования занимает в среднем на 0,2 секунды дольше, чем на кодирование. Наибольшая разница наблюдается при использовании файлов от 8 до 10 Кб. Для того чтобы наглядно показать расхождение в скорости преобразования данных была построена диаграмма, представленная на рисунке 19.

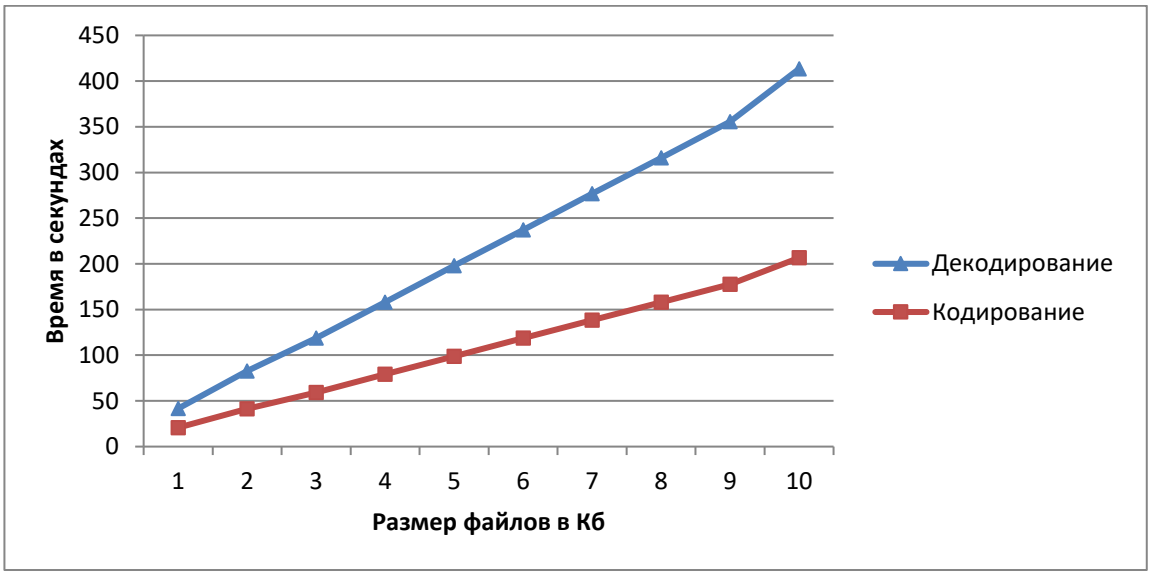

Рисунок 19 – Диаграмма изменения времени работы МК Atmega 328

На рисунке красной линией обозначен рост затраченного времени на декодирование, синей линией обозначен рост затраченного времени на кодирование информации. Время кодирования с учетом роста объема файла увеличивается в меньшей пропорциональности, чем увеличивается время на декодирование. Это связано с тем, что при обработке одного блока данных проводится одна операция мутации. В случае шифрования данных эта операция сводится к изменению одного случайного бита, при расшифровке, функция проводит поиск измененного бита и только потом его инкрементирует. Это ведет к дополнительным затратам времени на выполнение операции получения исходной информации. С увеличением объема файла, увеличивается количество обрабатываемых блоков, и как следствие разница во времени обработки данных.

Для сравнения было проведено тестирование на персональном компьютере с процессором Intel Core i3-3240, имеющим тактовую частоту 3,4 ГГц. Результаты исследования приведены в таблице 5.

Таблица 5 - Шифрование с помощью персонального компьютера

Размер файла в Кб	Кодирование в секундах	Декодирование в секундах
1	0,092	0,092
2	0,183	0,184
3	0,263	0,264
4	0,351	0,351
5	0,437	0,437
6	0,522	0,522
7	0,608	0,608
8	0,694	0,694
9	0,780	0,779
10	0,865	0,865

Из таблицы видно, что скорость шифрования данных составляет в среднем 10,8 Кб в секунду.

Процесс декодирования занимает в среднем на 0,0005 секунды дольше, чем на кодирование. Наибольшая разница наблюдается при использовании файлов от 9 до 10 Кб. Для того чтобы наглядно показать расхождение в скорости преобразования данных была построена диаграмма, представленная на рисунке 20.

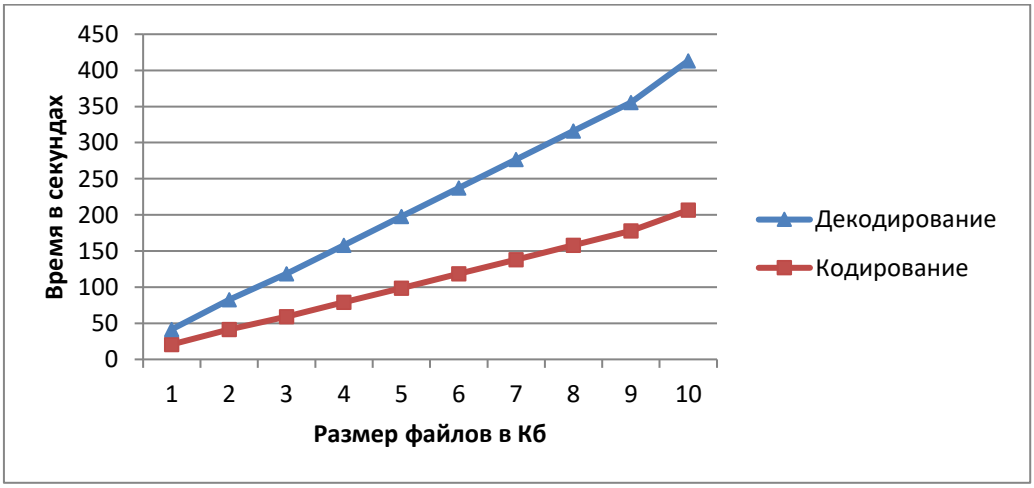

Рисунок 20 - Диаграмма изменения времени работы ПК

Из рисунка следует, что рост времени на декодирование с помощью персонального компьютера пропорционально равен росту времени на декодирование с помощью микроконтроллера.

Выявляя эффективность использования того или иного устройства были приведены следующие показатели:

- стоимость персонального компьютера, с помощью которого было проведено исследование – 56430 рублей;
- стоимость аппаратной части комплекса – 354 рублей, в которую входит плата Arduino Nano – 102,06 рублей, LCD 1602 дисплей – 125,97 рублей и модуль SD карты – 126,61 рублей;
- скорость шифрования данных с помощью МК – 49,6 байт в секунду;
- скорость шифрования данных с помощью ПК – 10,8 Кб в секунду.

Из этого следует что, стоимость персонального компьютера в 165,4 раза больше чем аппаратная часть комплекса. Обработка данных с помощью микроконтроллера производится в 212,5 раз медленнее. Общая эффективность использования персонального компьютера в 1,2 раза больше чем разработанного комплекса.

3.2 Целесообразность разработки с экономической точки зрения

Целесообразность разработки программно-аппаратного комплекса обуславливается полной автономией и энергонезависимостью системы. В условиях не позволяющих использовать стабильный источник питания или территориальной отдаленности от ЭВМ, когда существует необходимость зашифровать или расшифровать какую-либо информацию имеет место применение автономного комплекса. Низкая стоимость, малое энергопотребление и компактность являются основными преимуществами по

отношению к использованию персонального компьютера. Данные качества достигаются путем потери параметра скорости преобразования информации.

С экономической точки зрения данная разработка позволит сократить затраты на использование дорогостоящего оборудования в виде рабочих станций, программного обеспечения и больших объёмов электроэнергии.

Современные тенденции развития области применения генетических алгоритмов обуславливают расширение методик проектирования и разработку новых методов повышения их эффективности. Эти методики будут востребованы как при исследовании новых областей применения генетических алгоритмов, так и практическом применении для решения конечных задач. Предложенный подход решения задачи повышения эффективности генетических алгоритмов является одним из наиболее перспективных методов по повышению эффективности и расширению областей применения не только для рассмотренных задач генетического преобразования, но задач, для решения которых требуется значительные временные затраты.

Вывод по третьему разделу

Исходя из всего вышеизложенного, были выявлены преимущества использования программно-аппаратного комплекса перед персональным компьютером. В крупных организациях, использующих в качестве передаваемых данных большие объемы информации целесообразнее использовать стационарные рабочие станции в качестве шифровальных устройств. В случаях, если объем используемых для передачи данных ограничивается письменными извещениями, и нет возможности в покупке дорогостоящего оборудования, эффективнее всего использовать разработанный программно-аппаратный комплекс.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы был разработан программно-аппаратный комплекс кодирования цифровых данных с помощью генетического алгоритма для микроконтроллера Atmega 328.

В процессе разработки и проектирования решены следующие задачи:

- проведен анализ подходящих микроконтроллеров и методики кодирования данных с помощью генетического алгоритма;
- разработана структурная схема программно-аппаратного комплекса;
- разработано программное обеспечение низкого уровня для микроконтроллера, входящего в состав комплекса;
- исследована эффективность работы реализованного метода кодирования данных в разработанном программно-аппаратном комплексе.

В результате была достигнута основная цель работы: исключение из процесса шифрования данных на внешнем носителе необходимости применения ПК.

В будущем, возможны улучшения разработанной системы в следующих направлениях:

- реализация возможности шифрования каталогов файлов;
- размещение ключа шифрования на отдельном носителе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алехин, В.А. Микроконтроллеры PIC: основы программирования и моделирования в интерактивных средах MPLAB IDE, mikroC, TINA, Proteus. Практикум / В.А. Алехин. - М.: ГЛТ , 2016. - 248 с.

2. Бабаш, А. В. История криптографии. Часть I / А.В. Бабаш, Г.П. Шанкин. - М.: Гелиос АРВ, 2016. - 240 с.

3. Бабенко, Л. К. Современные алгоритмы блочного шифрования и методы их анализа / Л.К. Бабенко, Е.А. Ищукова. - М.: Гелиос, 2015. - 376 с.

4. Баззел, Р.Д. Информация и риск в маркетинге / Р.Д. Баззел, Д.Ф. Кокс, Р.В. Браун. - М.: Финстатинформ, 2015. - 758 с.

5. Баранов, В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы. Применение микроконтроллеров AVR: схемы, алгоритмы, программы / В.Н. Баранов. - М.: Додэка-XXI, 2006. - 288 с.

6. Баричев, С. Г. Основы современной криптографии. [Текст] / Баричев С.Г., Гончаров В.В., Серов О.Е.// - М.: Горячая линия-телеком, 3-е издание, 2011. - 175 с.

7. Белов, А.В. Микроконтроллеры AVR: от азов программирования до создания практических устройств / А.В. Белов. - СПб.: Наука и техника, 2016. - 544 с.

8. Борисов, М. А. Основы организационно-правовой защиты информации / М. А. Борисов, О. А. Романов // - М.: Книжный дом «ЛИБРОКОМ», 2012. - 208 с.

9. Брей, Б. Применение микроконтроллеров PIC 18. Архитектура, программирование и построение интерфейсов с применением C и ассемблера / Б. Брей. - СПб.: КОРОНА-Век, 2008. - 576 с.

10. Вельшенбах, М. Криптография на Си и C++ в действии. Учебное пособие / М. Вельшенбах. - М.: Триумф, 2014. - 462 с.

11. Горев, А. И. Обеспечение Информационной Безопасности / А Горев А И; Симаков А. - Москва: ИЛ**,** 2016**.** - 494 с.

12. Галицкий, А.В. Защита информации в сети - анализ технологий и синтез решений / А.В. Галицкий, С.Д. Рябко, В.Ф. Шаньгин. - М.: ДМК Пресс, 2013. - 615 с.

13. Жданов, О. Н. Методика выбора ключевой информации для алгоритма блочного шифрования / О.Н. Жданов. - М.: ИНФРА-М, 2015. - 869 с.

14. Евстифеев, А.В. Микроконтроллеры AVR семейства Classic фирмы ATMEL / А.В. Евстифеев. - М.: ДМК, 2015. - 286 с.

15. Евстифеев, А.В. Микроконтроллеры AVR фирмы ATMEL. Руководство пользователя / А.В. Евстифеев. - М.: ДМК, 2015. - 426 с.

16. Кадомцев, Б.Б. Динамика и информация / Б.Б. Кадомцев. - М.: [не указано], 2016. - 856 с.

17. Корсунов, Н.И. Модифицированный алгоритм шифрования данных [Текст]/ Н.И. Корсунов, А.И. Титов// Информационные системы и технологии. - №2 (64).- 2011. - 95с.

18. Малюк, А.А. Введение в защиту информации в автоматизированных системах. Учебное пособие / А.А. Малюк. - М.: Горячая линия - Телеком, 2016. - 148 с.

19. Мортон, Д. Микроконтроллеры AVR. Вводный курс / Д. Мортон. - М.: ДМК, 2015. - 272 с.

20. Мельников Защита информации в компьютерных системах / Мельников, Викторович Виталий. - М.: Финансы и статистика; Электроинформ, 2014. - 368 с.

21. Набока, Г. К. Разработка программно-аппаратного комплекса кодирования цифровых данных с помощью генетического алгоритма для микроконтроллера “Atmega 328” / Г. К. Набока. – Белгород, 20-я международная научно-практическая конференция Наука и образование: Отечественный и зарубежный опыт, 2019. – 391с.

22. Редькин, П.П. Микроконтроллеры Atmel архитектуры AVR32 семейства AT3 UC3 / П.П. Редькин. - М.: Техносфера, 2010. - 784 с.

23. Спесивцев, А.В. Защита информации в персональных ЭВМ / А.В. Спесивцев, В.А. Вегнер, А.Ю. Крутяков. - М.: Радио и связь, 2015. - 192 с.

24. Титов, А.И. Защита информации баз данных, хранящихся на удаленных серверах [Текст]/ А.И. Титов, Н.И. Корсунов // Журнал «Вопросы радиоэлектроники» , 2012. - 186 с.

25. Титов, А.И. Метод расширения ключа для кодирования информации, передаваемой по каналу связи[Текст] / А.И. Титов, Н.И. Корсунов, Муромцев В.В. // Научные ведомости БелГУ - №19(90) Выпуск 16/1, 2010. - 160 с.

26. Титов, А.И. Модифицированный алгоритм скрытного кодирования данных [Текст]/ А.И. Титов, Н.И. Корсунов // КНиТ - 2011 : труды Второй Международной научно-технической конференции, 3-7 октября 2011. – Белгород.: Изд-во НИУ БелГУ, 2011. - 154 с.

27. Титов, А.И. Модифицированный блочно-итерационный метод шифрования и дешифрования данных [Текст]/ А.И. Титов, Н.И. Корсунов // Информационные системы и технологии, - № 1(69) январь-февраль, 2012. - 114 с.

28. Титов, А.И. Обеспечение скрытности кодирования данных помехоустойчивыми кодами[Текст]/ А.И. Титов, Н.И. Корсунов // Научные ведомости БелГУ, Серия История Политология Экономика Информатика, № 8(151)Выпуск 14/1, 2013. - 117 с.

29. Титов, А.И. Повышение эффективности защиты информации модификацией шифра Вижинера [Текст]/ А.И. Титов, Н.И. Корсунов // Научные ведомости БелГУ, Серия История Политология Экономика Информатика, № 7(78)Выпуск 14/1, 2010. - 175 с.

30. Титов, А.И. Применение генетических алгоритмов в эволюционном методе кодирования (шифрования) данных [Текст]/А.И. Титов//Сборник трудов международной молодежной конференции. Прикладная математика, управление и информатика, 2012. - 601 с.

31. Титов, А.И. Эволюционное кодирование данных [Текст]/ А.И. Титов // Журнал «Вопросы радиоэлектроники» , 2013. - 96 с.

32. Титов, А.И. Эволюционные методы кодирования данных, пример работы алгоритмов кодирования [Текст]/ А.И. Титов, Н.И. Корсунов, К.И. Логачев // Научные ведомости БелГУ, Серия История Политология Экономика Информатика, № 1(120)Выпуск 14/1, 2012. - 126 с.

33. Хартов, В.Я. Микроконтроллеры AVR / В.Я. Хартов. - М.: МГТУ , 2011. - 240 с.

34. Хартов, В.Я. Микроконтроллеры AVR. Практикум для начинающих: Учебное пособие / В.Я. Хартов. - М.: МГТУ им. Баумана, 2012. - 280 с.

35. Хоффман, Л. Дж. Современные методы защиты информации / Л.Дж. Хоффман. - Москва**:** СПб. [и др.] : Питер, 2014. - 264 с.

36. Шаньгин, В. Ф. Информационная безопасность и защита информации / В.Ф. Шаньгин. - Москва: Огни, 2016. - 551 с.

ПРИЛОЖЕНИЕ А

Листинг программы:

```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <iarduino_RTC.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);

File dataFile1;
File dataFile;
File FileKey;

const int chipSelect = 4;
char buff;
int f = 0;
byte result1 = B00000000;

int x,y,q,w;
int key;
int count=0;
int a=0;
int b=0;
int d=0;
int C11, C12, C21, C22, D11, D12, D21, D22;

unsigned long timeStart, timeFinish, timeRun;

String gen1;
String gen2;
String str = "EXAMPLE.txt";
String str1 = "EXAMPL1.txt";
String str2 = "EXAMPL2.txt";
```

```
void setup() {
  Serial.begin(9600);

  pinMode(2, INPUT);
  pinMode(3, INPUT);
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);

  lcd.init();
  lcd.backlight();
  lcd.clear(); lcd.setCursor(0,0); lcd.print("Hello");

     while (!Serial) {}
  lcd.clear(); lcd.setCursor(0,0); lcd.print("Initializing");
  lcd.setCursor(0,1); lcd.print("SD card...");

  if (!SD.begin(chipSelect)) {
    lcd.clear(); lcd.setCursor(0,0); lcd.print("Card failed");
    lcd.setCursor(0,1); lcd.print("or not present");
    return;}

  lcd.clear(); lcd.setCursor(0,0); lcd.print("Card");
  lcd.setCursor(0,1); lcd.print("initialized");}

void loop() {

  RTC.begin();
  int but2 = digitalRead(2);
  int but3 = digitalRead(3);

  if((but2 == HIGH)&&(b == 1)){
    b = 0;}
    if((but2 == LOW)&&(b == 0)){
      b = 1;
      gen1 = str;
      gen2 = str1;
      count = 1;
      lcd.clear(); lcd.setCursor(0,0); lcd.print("crypt");}
```

```
    if((but3 == HIGH)&&(d == 1)){
      d = 0;}
      if((but3 == LOW)&&(d == 0)){
        d = 1;
        gen1 = str1;
        gen2 = str2;
        count = 2;
        lcd.clear(); lcd.setCursor(0,0); lcd.print("decrypt");}
    delay(500);

     if(count==1){

  timeStart = millis();

  DateTime now = RTC.now();
  randomSeed(now.second(), DEC);
  key = randomSeed;
  FileKey = SD.open("Key.txt");
  if(FileKey){key = FileKey.read();}
  FileKey.close();

  dataFile = SD.open(gen1);
  dataFile1 = SD.open(gen2, FILE_WRITE);

  if (dataFile) {
   while (dataFile.available()) {
      x = dataFile.read();
      y = dataFile.read();
      q = dataFile.read();
      w = dataFile.read();

a = bitRead(key, 7);
if(a==0) {C11=x; C21=y;}
else {C11=y; C21=x;}
a = bitRead(key, 6);
if(a==0) {C12=q; C22=w;}
else {C12=w; C22=q;}

D11=C11^C21;
```

```
D12=C12^C22;

a = bitRead(key, 5);
if(a==0) {D21=C12; D22=C21;}
else {D21=C11; D22=C22;}

a = bitRead(key, 4);
if(a==0) {C11=D11; C21=D12;}
else {C11=D12; C21=D11;}
a = bitRead(key, 3);
if(a==0) {C12=D21; C22=D22;}
else {C12=D22; C22=D21;}

D11=C11^C21;
D12=C12^C22;

a = bitRead(key, 2);
if(a==0) {D21=C12; D22=C21;}
else {D21=C11; D22=C22;}

if (dataFile1)
{
  buff = (char) D11;
  dataFile1.print(buff);
  buff = (char) D12;
  dataFile1.print(buff);
  buff = (char) D21;
  dataFile1.print(buff);
  buff = (char) D22;
  dataFile1.print(buff);

  lcd.setCursor(0,0); lcd.print("The process");
  lcd.setCursor(0,1); lcd.print("is running");
  Serial.print(buff);}}}
  else {
     lcd.clear(); lcd.setCursor(0,0); lcd.print("Error opening");
     lcd.setCursor(0,1); lcd.print("file");}
dataFile.close();
dataFile1.close();
```

```
timeFinish = millis();
timeRun=timeFinish-timeStart;

lcd.clear(); lcd.setCursor(0,0); lcd.print("Successfully");lcd.setCursor(0,1);
lcd.print(timeRun);
count = 0;}

            if(count==2){

timeStart = millis();

  FileKey = SD.open("Key.txt");
  if(FileKey){key = FileKey.read();}
  FileKey.close();

dataFile = SD.open(gen1);
  dataFile1 = SD.open(gen2, FILE_WRITE);

  if (dataFile) {
   while (dataFile.available()) {
      x = dataFile.read();
      y = dataFile.read();
      q = dataFile.read();
      w = dataFile.read();

a = bitRead(key, 2);
if(a==0) {C12=q; C21=w; C11=x^C21; C22=y^C12;}
else {C11=q; C22=w; C12=y^C22; C21=x^C11;}

a = bitRead(key, 3);
if(a==0){D11=C11; D12=C21;}
else{D11=C21; D12=C11;}
a = bitRead(key, 4);
if(a==0){D21=C12; D22=C22;}
else{D21=C22; D22=C12;}

a = bitRead(key, 5);
if(a==0) {C12=D21; C21=D22; C11=D11^C21; C22=D12^C12;}
```

```
else {C11=D21; C22=D22; C12=D12^C22; C21=D11^C11;}

a = bitRead(key, 6);
if(a==0){D11=C11; D12=C21;}
else{D11=C21; D12=C11;}
a = bitRead(key, 7);
if(a==0){D21=C12; D22=C22;}
else{D21=C22; D22=C12;}

if (dataFile1)
{
  buff = (char) D12;
  dataFile1.print(buff);
  buff = (char) D11;
  dataFile1.print(buff);
  buff = (char) D22;
  dataFile1.print(buff);
  buff = (char) D21;
  dataFile1.print(buff);

  lcd.setCursor(0,0); lcd.print("The process");
  lcd.setCursor(0,1); lcd.print("is running");
  Serial.print(buff);
  }}}
  else {
     lcd.clear(); lcd.setCursor(0,0); lcd.print("Error opening");
     lcd.setCursor(0,1); lcd.print("file");}
dataFile.close();
dataFile1.close();

timeFinish = millis();
timeRun=timeFinish-timeStart;

lcd.clear(); lcd.setCursor(0,0); lcd.print("Successfully");lcd.setCursor(0,1);
lcd.print(timeRun);
count = 0;}
}
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) *(Ф.И.О.)*